An Exact Minimization of AND-EXOR Expressions Using Reduced Covering Functions(Fundamental Studies on Computational Complexity)

AUTHOR(S):

SASAO, Tsutomu

数理解析研究所講究録
第 876 巻 1994 年 50-59

An Exact Minimization of AND-EXOR Expressions Using Reduced Covering Functions

Tsutomu SASAO 笹尾 勤
Department of Computer Science and Electronics
Kyushu Institute of Technology, Iizuka 820, Japan

August 9, 1993

Abstract

This paper considers a method to derive an EXOR sum-of-products expression (ESOP) having the minimum number of products for a given logic function. The minimization method uses a reduced covering function, which is an improvement of the method proposed by Perkowski and Chrzanowska-Jeske. Binary Decision Diagrams (BDDs) are used to obtain exact solutions. Various techniques to reduce computation time and memory storage are developed. Experimental results for functions with up to 9 variables are shown.

I Introduction

EXOR sum-of-products expressions (ESOPs) are obtained by EXORing arbitrary products. ESOPs have several advantages over sum-of-products expressions (SOPs) [23]. The most important one is that ESOP realizations are often less expensive than SOP realizations. For example, to represent arbitrary function of four variables, ESOPs require, on the average, 3.66 products while SOPs require 4.13 products [25]. We conjecture that this is true for an arbitrary n. We have also proved that an arbitrary function of n variables can be realized with at most 2^{n-2} products when $n \geq 6$, while SOPs require at most 2^{n-1} products [25]. In addition, we demonstrated that ESOPs require fewer products and fewer literals than SOPs to represent arithmetic functions and other functions [24]. An ESOP requires at most $2 \cdot 3^r$ products to realize an arbitrary n-variable symmetric function, where $n = 2r$. For any symmetric function, an ESOP requires no more products than an SOP [23, 20]. Because many of the arithmetic functions have symmetrical properties, ESOPs are useful for arithmetic circuits. In most technologies, EXOR gates are more expensive than OR gates. However, even if we assume that the cost of a 2-input EXOR is twice as expensive as a 2-input NOR, the EXOR based circuits are more economical than ones based on only ANDs and ORs [24]. Also, in LUT based FPGAs [21], ORs and EXORs have exactly the same cost and the same propagation delay.

In the *ESOP minimization problem*, one seeks an ESOP having the minimum number of products. Many papers have considered this problem [6, 18, 16, 23, 22, 7, 9, 2]. Recently, Perkowski and Chrzanowska-Jeske[19] formulated the problem by using a Helliwell equation. Their formulation is to find a 3^n bit vector with the minimum weight satisfying the Helliwell equation, where n is the number of the variables in the given logic function. They also presented various methods to solve this problem. However, the computational complexity of their methods are $O(2^{3^n})$ because they consider most of the combinations of the 3^n bit vectors. Thus, the order of the complexity is the same as in exhaustive search. However, their formulation has provided insights into exact ESOP minimization.

In this paper, we present an improved method for ESOP minimization by using reduced covering function and BDDs.

II Minimization of SOPs and ESOPs

2.1 Minimization of SOPs

Suppose that we want to obtain a minimum SOP representing an n-variable function f. Assume we are given the set of true minterms (1-cells in Karnaugh map) and the set of all prime implicants (PIs)

for f. Let there be ξ PIs for f. Any SOP of f is a subset of PIs that covers all true minterms. For some SOP for f, let g_j be a logic variable that is 1 iff the j-th PI is in this SOP. Thus, the problem of finding a minimal SOP is identical to the problem of finding an assignment for the g_j's such that the corresponding SOP represents f and the weight $\sum_{i=0}^{\xi-1} g_j$ is minimum. We derive such an assignment by using the *Petrick Equation* defined as follows:

$$P(g_0, g_1, \cdots, g_{\xi-1}) = \bigwedge_{i=0}^{N-1} S_i,$$

where $S_i = \vee_{g_j \in T_i} g_j$, N is the number of true minterms in f, and T_i represents the set of PIs that covers the i-th minterm. S_i is 1 iff the i-th true minterm is covered by at least one PI, and $P(g_0, g_1, \cdots, g_{\xi-1})$ is 1 iff every minterm is covered by at least one PI[17].

A solution to the SOP minimization problem is an assignment of 0's and 1's to g_i that satisfies P with the fewest 1's.

Example 2.1 *Let us derive the minimum SOPs for the function f shown in Fig. 2.1 by using the Petrick equation. Note that f has 6 minterms and 6 PIs, which are shown in Figs. 2.2 and 2.3. For*

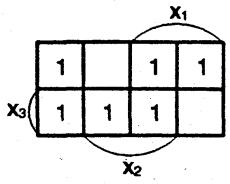


Figure 2.1:

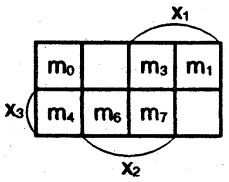
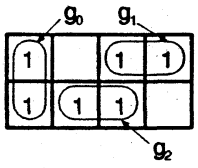


Figure 2.2: Minterms



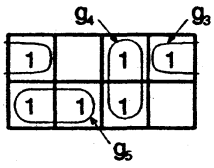


Figure 2.3: PIs

these minterms, we have the following relations:

$$\begin{aligned}
m_0 : S_0 &= g_0 \vee g_3, \\
m_1 : S_1 &= g_1 \vee g_3, \\
m_2 : S_2 &= g_1 \vee g_4, \\
m_3 : S_3 &= g_0 \vee g_5, \\
m_4 : S_4 &= g_2 \vee g_5, \text{ and} \\
m_5 : S_5 &= g_2 \vee g_4.
\end{aligned}$$

All the minterms are covered iff $S_i = 1$ $(i = 0, \cdots, 5)$. *This is true iff* $P(g) = 1$, *where*

$$P(g) = (g_0 \vee g_3)(g_1 \vee g_3)(g_1 \vee g_4)(g_0 \vee g_5)(g_2 \vee g_5)(g_2 \vee g_4).$$

Two assignments satisfying $P(g) = 1$ *with minimum weights are* $g_0 = g_1 = g_2 = 1$ *and* $g_3 = g_4 = g_5 = 1$. *One can find them by expanding* $P(g)$ *into an SOP form. Thus, the corresponding minimum SOPs for f are* $\bar{x}_1\bar{x}_2 \vee x_1\bar{x}_3 \vee x_2x_3$ *and* $\bar{x}_2\bar{x}_3 \vee x_1x_2 \vee \bar{x}_1x_3$.

2.2 ESOP minimization problem

In the case of ESOP minimization, we have to consider all the products instead of just the PIs. Also, the covering problem is an odd-even type, since $1 \oplus 1 = 0$. This problem corresponds to finding a minimum cost solution of the *Helliwell Equation* $H(g) = 1$, where $H(g)$ is a product-of-EXOR sums expression defined as follows:

$$H(g_0, g_1, \cdots, g_{\xi-1}) = \bigwedge_{i=0}^{N-1} S_i,$$

where $S_i = \sum_{g_j \in T_i} \oplus g_j \oplus f(a_i) \oplus 1$, N is the total number of the cells in the Karnaugh map $(= 2^n)$, and ξ is the number of all possible products $(= 3^n)$. $g_j = 1$ iff the j-th product is contained in the ESOP. S_i shows the condition that each true minterm is covered by products an odd number of times, and each false minterm (0-cell in Karnaugh map) is covered by products an even number of times. $H(g_0, g_1, \cdots, g_{\xi-1})$ shows that products cover true minterms an odd number of times and false minterms an even number of times. Thus, the number of variables in the Helliwell function is 3^n.

Example 2.2 *Let us derive the minimum ESOPs for the function f shown in Fig. 2.4 by using the Helliwell equation. Note that $N = 4$ is the number of cells in the map, and $\xi = 9$ is the number of loops in Fig. 2.6. For each cell, we have*

$$\begin{aligned} m_0 : S_0 &= g_0 \oplus g_2 \oplus g_6 \oplus g_8 \oplus 1 \oplus 1, \\ m_1 : S_1 &= g_1 \oplus g_2 \oplus g_7 \oplus g_8 \oplus 0 \oplus 1, \\ m_2 : S_2 &= g_3 \oplus g_5 \oplus g_6 \oplus g_8 \oplus 0 \oplus 1, \text{ and} \\ m_3 : S_3 &= g_4 \oplus g_5 \oplus g_7 \oplus g_8 \oplus 1 \oplus 1. \end{aligned}$$

True minterms are covered by loops an odd number of times and false minterms are covered by loops

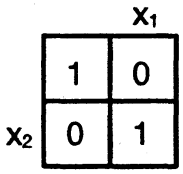


Figure 2.4:

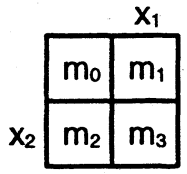


Figure 2.5:

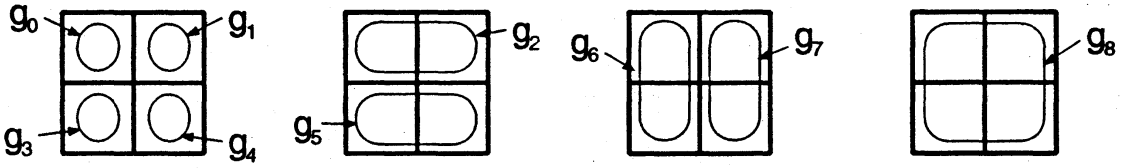


Figure 2.6: Products

an even number of times iff $S_i = 1$ $(i = 0, 1, 2, 3)$. This is true iff the Helliwell equation is $H(g) = 1$, where

$$\begin{aligned} H(g) = {} & (g_0 \oplus g_2 \oplus g_6 \oplus g_8 \oplus 1 \oplus 1) \cdot (g_1 \oplus g_2 \oplus g_7 \oplus g_8 \oplus 0 \oplus 1) \cdot \\ & (g_3 \oplus g_5 \oplus g_6 \oplus g_8 \oplus 0 \oplus 1) \cdot (g_4 \oplus g_5 \oplus g_7 \oplus g_8 \oplus 1 \oplus 1). \end{aligned}$$

Assignments that make $H(g) = 1$ with minimum weights are $g_0 = g_4 = 1$, $g_2 = g_7 = 1$ or $g_5 = g_6 = 1$. The corresponding minimum ESOPs are $x_1x_2 \oplus \bar{x}_1\bar{x}_2, \bar{x}_2 \oplus x_1$ and $x_2 \oplus \bar{x}_1$, respectively.

[19] presented various methods to find a minimum solution for the Helliwell equation, but did not show any computational results. To evaluate the usefulness of their methods, we implemented a similar method and confirmed that "the methods are very time and memory consuming" [19].

III Reduced Covering Function

This section presents a new method for exact ESOP minimization by using reduced covering functions. This function involves $2^r \cdot 3^{n-r}$ variables, and requires fewer variables than the Helliwell function. Let f be an n-variable function $(n \geq 5)$, $B = \{0, 1\}$, and $T = \{0, 1, 2\}$. Let $X = (x_1, x_2, \cdots, x_n)$ be a vector of binary variables, and $X = (X_1, X_2)$ be a partition of X. Let $(n - r)$ be the number of variables in X_1, and r be the number of variables in X_2, where $r = 0, 1, 2, 3$, or 4. Then, f can be represented as

$$f(X_1, X_2) = \sum_{\boldsymbol{a}} \oplus X_1^{\boldsymbol{a}} \cdot g(\boldsymbol{a} : X_2), \tag{3.1}$$

where $\boldsymbol{a} \in T^{n-r}$,

$$X_1^{\boldsymbol{a}} = x_1^{a_1} \cdot x_2^{a_2} \cdot \cdots \cdot x_{n-r}^{a_{n-r}}, \quad x_i^{a_i} = \begin{cases} \bar{x}_i & a_i = 0 \\ x_i & a_i = 1, \\ 1 & a_i = 2 \end{cases}$$

and $g(\boldsymbol{a}: X_2)$ is an r-variable function.

By assigning a constant $\boldsymbol{b} \in B^{n-r}$ to X_1 in (3.1), we have

$$f(\boldsymbol{b}, X_2) = \sum_{\boldsymbol{c} \geq_R \boldsymbol{b}} \oplus g(\boldsymbol{c}: X_2), \tag{3.2}$$

where $\sum \oplus$ denotes the EXOR with respect to $\boldsymbol{c} \in T^{n-r}$ satisfying $\boldsymbol{c} \geq_R \boldsymbol{b}$. The symbol $\geq_R$ denotes the binary relation $\{(0,0),(1,1),(2,2),(2,0),(2,1)\}$. For each $\boldsymbol{b}$, there are 2^{n-r} different $\boldsymbol{c}$ that satisfy $\boldsymbol{c} \geq_R \boldsymbol{b}$. Because there are 2^{n-r} different $\boldsymbol{b}$, we have 2^{n-r} different equations of the form (3.2). Next, by assigning a constant $\boldsymbol{d} \in B^r$ to X_2 in (3.2), we have

$$f(\boldsymbol{b}, \boldsymbol{d}) = \sum_{\boldsymbol{c} \geq_R \boldsymbol{b}} \oplus g(\boldsymbol{c}: \boldsymbol{d}). \tag{3.3}$$

There are 2^{n-r} different $\boldsymbol{b}$ and 2^{n-r} different $\boldsymbol{d}$, so we have $2^{n-r} \cdot 2^r = 2^n$ different equations of form (3.3). Note that $g(\boldsymbol{c}: \boldsymbol{d})$ is either 0 or 1. (3.3) holds for all possible combinations of $\boldsymbol{b}$ and $\boldsymbol{d}$ at the same time if and only if $R(g) = 1$, where

$$R(g) = \bigwedge_{(\boldsymbol{b},\boldsymbol{d})} ([\sum_{\boldsymbol{c} \geq_R \boldsymbol{b}} g(\boldsymbol{c}: \boldsymbol{d})] \oplus f(\boldsymbol{b}, \boldsymbol{d}) \oplus 1) \tag{3.4}$$

and $\bigwedge_{(\boldsymbol{b},\boldsymbol{d})}$ denotes the logical product with respect to all possible $\boldsymbol{b} \in B^{n-r}$ and $\boldsymbol{d} \in B^r$. $R(g)$ is called a *Reduced Covering Function* (RCF).

Let $g(\boldsymbol{c}: \boldsymbol{d})$ be Boolean variables, where $\boldsymbol{c} \in T^{n-r}$ and $\boldsymbol{b} \in B^r$. Then, the total number of variables in RCF is $2^r \cdot 3^{n-r}$. An assignment for $g(\boldsymbol{c}, \boldsymbol{d})$ that satisfies $R(g) = 1$ is called a *solution* of $R(g) = 1$.

A minimum ESOP for f can be written in the form (3.1). Let $g(\boldsymbol{a}: X_2)$ be an r-variable function. We want to obtain 3^{n-r} such functions. Because $g(\boldsymbol{a}: X_2)$ can be written as

$$g(\boldsymbol{a}: X_2) = \sum_{\boldsymbol{d} \in B^r} \oplus X_2^{\boldsymbol{d}} g(\boldsymbol{a}: \boldsymbol{d}),$$

$g(\boldsymbol{a}: X_2)$ can be obtained from the set of $g(\boldsymbol{c}: \boldsymbol{d})$ that satisfy the RCF. A minimum ESOP corresponds to the solution of the RCF with the minimum value of the cost function:

$$\sum_{\boldsymbol{a}} \tau(g(\boldsymbol{a}: X_2)), \tag{3.5}$$

where $\sum_{\boldsymbol{a}}$ denotes the arithmetic sum with respect to all possible $\boldsymbol{a} \in T^{n-r}$, and $\tau(g)$ denotes the number of products in a minimum ESOP for g. Thus, we have the following theorem.

Theorem 3.1 *A minimum ESOP of an n-variable function corresponds to an assignment of the RCF having a minimum cost of (3.5).*

Because we have a table for minimum ESOPs for up to 4-variables, the values of $\tau(g(\boldsymbol{a}: X_2))$ for $r = 0, 1 \cdots$, and 4 are available.

Cost functions

0) When $r = 0$. $g(\boldsymbol{a}: X_2) = g(\boldsymbol{a})$ are 0-variable functions (constants), and do not depend on X_2. There are 3^n different $g(\boldsymbol{a})$, and they correspond to the 3^n different products of n variables. In this case, the RCF is the same as the Helliwell function [19]. The cost function is

$$\tau(g(\boldsymbol{a}: X_2)) = g(\boldsymbol{a}).$$

1) When $r = 1$. $g(\boldsymbol{a}: X_2)$ are 1-variable functions and can be represented as

$$g(\boldsymbol{a}: X_2) = X_2^0 \cdot g(\boldsymbol{a}: 0) \vee X_2^1 \cdot g(\boldsymbol{a}: 1).$$

An ESOP uses at most one product to represent $g(\boldsymbol{a}: X_2)$, and requires a product only when $g(\boldsymbol{a}: 0) \vee g(\boldsymbol{a}: 1) = 1$. Thus, the cost function is

$$\tau(g(\boldsymbol{a}: X_2)) = g(\boldsymbol{a}: 0) \vee g(\boldsymbol{a}: 1).$$

2) When $r = 2$. $g(\boldsymbol{a} : X_2) = g(u, v)$ are 2-variable functions. An ESOP uses at most two products to represent $g(\boldsymbol{a} : X_2)$, and the cost function is

$$\tau(g(\boldsymbol{a} : X_2)) = \begin{cases} 0 & \text{when } g(u,v) = 0. \\ 1 & \text{when } g(u,v) = 1, \bar{u}, u, \bar{v}, v, \bar{u} \cdot \bar{v}, \\ & \qquad \bar{u} \cdot v, u \cdot \bar{v}, \text{ or } u \cdot v \\ 2 & \text{otherwise.} \end{cases}$$

3) When $r = 3$. $g(\boldsymbol{a} : X_2)$ are 3-variable functions. An ESOP uses at most three products to represent $g(\boldsymbol{a} : X_2)$. The cost function is complex, but it is possible to represent by logic functions.

4) When $r = 4$. $g(\boldsymbol{a} : X_2)$ are 4-variable functions. An ESOP uses at most 6 products. The cost function is complex, but it is possible to represent by logic functions.

Example 3.1 *Let us obtain minimum ESOPs for the function in Fig. 2.4 by using RCF. Let* $X_1 = (x)$ *and* $X_2 = (y)$ *be the partition of* $X = (x, y)$. *In this case, we obtain the minimum ESOP having the following form:*

$$f(x, y) = \bar{x} \cdot g(0 : y) \oplus x \cdot g(1 : y) \oplus 1 \cdot g(2 : y).$$

Now, we will find three functions such that $\sum_{i=0}^{2} \tau(g(i : y))$ *is minimum. Because* $g(i : y)$ *can be expanded as*

$$g(i : y) = \bar{y} \cdot g(i : 0) \oplus y \cdot g(i : 1),$$

we have the following expression:

$$f(x, y) = \bar{x} \cdot \bar{y} \cdot g(0 : 0) \oplus \bar{x} \cdot y \cdot g(0 : 1) \oplus x \cdot \bar{y} \cdot g(1 : 0) \oplus x \cdot y \cdot g(1 : 1) \oplus 1 \cdot \bar{y} \cdot g(2 : 0) \oplus 1 \cdot y \cdot g(2 : 1).$$

By assigning (0,0), (0,1), (1,0) and (1,1) into (x,y), we have four equations:

$$\begin{aligned} f(0,0) &= g(0:0) \oplus g(2:0) = 1, \\ f(0,1) &= g(0:1) \oplus g(2:1) = 0, \\ f(1,0) &= g(1:0) \oplus g(2:0) = 0, \text{ and} \\ f(1,1) &= g(1:1) \oplus g(2:1) = 1. \end{aligned}$$

The RCF is

$$\begin{aligned} R(g) = {} & [g(0:0) \oplus g(2:0)][g(0:1) \oplus g(2:1) \oplus 1] \cdot \\ & [g(1:0) \oplus g(2:0) \oplus 1][g(1:1) \oplus g(2:1)]. \end{aligned}$$

The minimum assignments that make $R(g) = 1$ *true are*

$$\begin{aligned} g(0:0) &= g(1:1) = 1, \\ g(1:0) &= g(1:1) = g(2:0) = 1, \text{ and} \\ g(0:0) &= g(0:1) = g(2:1) = 1. \end{aligned}$$

The corresponding minimum ESOPs are $x_1 x_2 \oplus \bar{x}_1 \bar{x}_2, x_1 \oplus \bar{x}_2$ *and* $\bar{x}_1 \oplus x_2$, *respectively. Fig. 3.1 shows the role of the variables. In the case of RCF, the sets of variables represent logic functions.*

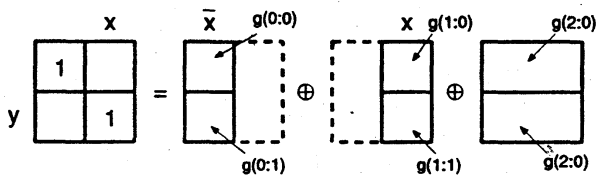


Figure 3.1: The role of variables.

IV Optimization Using Binary Decision Diagrams.

A minimum ESOP for a given function corresponds to an assignment for RCF with the minimum cost. In this part, we show a method to find such assignments by using Binary Decision Diagrams (BDDs) [3, 15]. Let $H(g) = 1$ be a Boolean equation showing the constraints of an optimization problem, where $g = (g_0, g_1, \cdots, g_{\xi-1})$. For example, $H(g) = 1$ can be the Petrick equation or the Helliwell equation. Assignments satisfying $H(g) = 1$ are called *feasible solutions*. When $H(g)$ is represented by a BDD, a path from the root node to a constant 1 node corresponds to a feasible solution. Such a path is called a *1-path* of a BDD. By attaching a cost to the set of edges, we can make the paths represent the cost of the solution. Thus, we can convert the optimization problem into the shortest path problem [13]. The BDD for $H(g)$ is usually too large to build, so we need various techniques to reduce the size of the BDD.

Example 4.1 *Fig. 4.1 is the BDD for Helliwell function in Example 2.2. Fig. 4.2 is the BDD for RCF in Example 3.1. Note that RCF requires fewer nodes than the Helliwell function.*

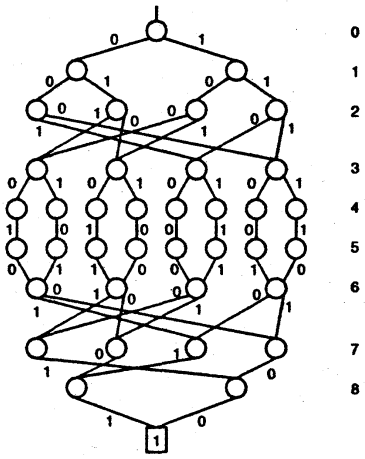


Figure 4.1: BDD for Helliwell function

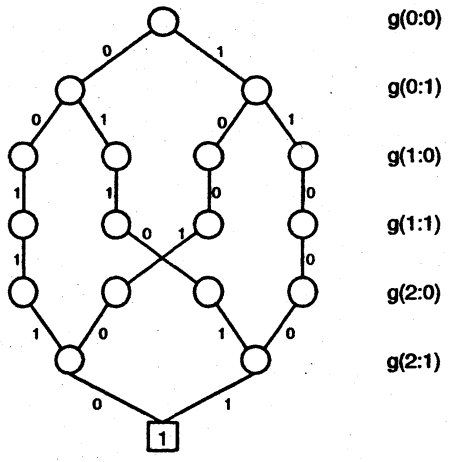


Figure 4.2: BDD for RCF

V Number of Variables for RCF.

This part considers the number of variables of a RCF for the optimization of an ESOP with n variables. Suppose that we use the table of minimum ESOPs with r variables. Then, the number of variables in RCF is $\xi(n, r) = 2^r \cdot 3^{n-r}$. Table 5.1 shows the number of variables for RCFs. It shows that as r increases, $\xi(n, r)$ decreases. However, the computation time for cost functions and $U(t_1)$ will increase as r increases. Note that the number of the variables in the RCF does not always show the complexity of the problem.

VI Various Techniques to Reduce Computation Time and Memory Requirement.

The BDD formulation of the problem is straightforward. However, the BDDs so formed are usually very large. In general, almost all the computation time is spent in the generation of BDDs, and the CPU time for finding the shortest path is relatively small. Thus, the problem is how to generate the BDD efficiently.

6.1 Lower bound on the number of products.

Although we have various ways to reduce the computation time and memory requirement for the construction of BDDs, the best way is not to construct a BDD. We have very good heuristic minimization

Table 5.1: Number of the variables for RCF

	$\xi(n,r)$				
n	$r=0$	$r=1$	$r=2$	$r=3$	$r=4$
3	27	18	12		
4	81	54	36	24	
5	243	162	108	72	48
6	729	486	324	216	144
7	2187	1458	972	648	432
8	6561	4374	2916	1944	1296
9	19683	13122	8748	5832	3888

program EXMIN2, which finds near optimum solutions quickly. Suppose we know that the ESOP for a function f requires at least t_2 products, and EXMIN2 produced an ESOP with t_1 products. If $t_1 = t_2$, then the solution obtained by EXMIN2 is a minimum, and we can stop the procedure without generating the BDD. We have a method to obtain the lower bound on the number of products in ESOPs.

Theorem 6.1 *Let the function f be expanded as $f = \bar{x}_i \cdot f_{i_0} \oplus x_i \cdot f_{i_1}$. Then, the ESOP for f contains at least $max\{L1, L2\}$ products, where*

$$L_1 = \frac{1}{2} \max_{i=1}^{n} \{\tau(f_{i_0}) + \tau(f_{i_1}) + \tau(f_{i_2})\}, \text{ and}$$
$$L_2 = 1 + \min_{q_j} \{L_1(q_j \oplus f)\}$$

$f_{i_2} = f_{i_0} \oplus f_{i_1}$, q_j represents a product containing at least one minterm of f, and $\tau(g)$ is the number of the products in a minimum ESOP for g.

To use the above theorem, we have to know the number of the products in the minimum ESOPs for $(n-1)$ variable functions. Up to $n = 6$, we can do this by a table look-up method [12].

6.2 Upper bound on the number of products.

EXMIN2 is a heuristic ESOP simplification algorithm and produces near optimum solutions [26]. Let t_1 be the cost of a near optimum solution of EXMIN2, and let $U(t_1)$ be the logic function showing that the cost is less than t_1.

$$U(t_1) = \begin{cases} 0 & (\sum \tau(g(\boldsymbol{a} : X_2)) \geq t_1 \\ 1 & \text{otherwise.} \end{cases}$$

We can generate the BDD for $R(g) \cdot U(t_1)$, instead of for $R(g)$. This will drastically reduce the BDD size and computation time. $R(g) \cdot U(t_1)$ is called a *Modified Reduced Covering Function* (MRCF).

Example 6.1 *Fig. 6.1 is the BDD for $H(g) \cdot U(3)$, which is the modified RCF with $r = 0$. This BDD has only three* 1*-paths.*

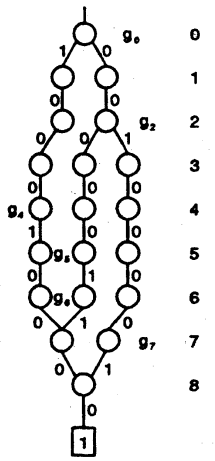


Figure 6.1: BDD for $H(g) \cdot U(3)$

6.3 Methods for multiplication.

To generate the BDD for MRCF, we have to multiply 2^n parity functions. For each multiplication of a parity function, the number of nodes in the BDD tends to be double. Thus, we often encounter the memory overflow errors during the generation of BDDs. To avoid such errors, we carefully choose the order of the multiplication. Currently, we use the order that increases the least number of variables in the BDD for each step.

6.4 Zero-suppressed BDD.

A MRCF represents a set of vectors that satisfy the RCF. Although the number of variables is $O(3^n)$, the number of non-zero elements in the vectors is $O(2^n)$, because $t_1 \leq 2^n$ and the weight is less than t_1. Thus, MRCF represents a set of vectors with small weight. Zero-suppressed BDDs [15] are a variant of BDDs, and represent such sets very efficiently. By using zero-suppressed BDDs, we can reduce the computation time and memory requirement significantly.

VII Minimization Algorithm.

Step 1. Let F be an ESOP for f simplified by EXMIN2.
$t_1 \leftarrow \tau(F)$,
$t_2 \leftarrow$ Lower bound obtained by Theorem 6.1.

Step 2. If $t_1 = t_2$, then F is the minimum and stop.

Step 3. Construct the BDD for $R(g) \cdot U(t_1)$.
If the BDD represents the constant 0 function, then F is the minimum and stop.

Step 4. Find a shortest path.

Step 5. Construct the ESOP corresponding to the path.

VIII Experimental Results

We coded the algorithm in C, and implementing it on an HP9000 Model 720 workstation with 64 Mega-byte main memory. We used two types of BDD packages, one is based on the conventional BDD [1] and the other is based on zero-suppressed BDD [15]. Neither of them use complemented edges [14]. In addition, we developed code to generate the BDDs for bounding functions. Upper and lower bounds on the number of the products in ESOPs are obtained by separate programs [26, 12].

8.1 With Helliwell functions.

Up to $n = 3$, we could easily build BDDs of Helliwell functions. Thus, the minimization of 3-variable functions was easy. However, BDDs of Helliwell functions for $n = 4$ involve $3^4 = 81$ variables and were very expensive.

8.2 With Modified Reduced Covering Functions.

The BDDs of MRCF for $n = 5$ were easy to derive, and we could minimize all the 5-variable functions that we tried. The most complicated 5 variable function required 9 products, and EXMIN2 produced these solutions. However, up to $n = 5$, we can derive the minimum ESOPs by table look up: it requires only 6 seconds on the average [12]. For $n = 9$, we could minimize the ESOP with at most 4 products. Table 8.1 shows the cpu time and memory requirement using zero-suppressed BDD. For large problems, conventional BDDs required more cpu time, and more memory than zero-suppressed BDDs.

IX Conclusion

In this paper, we presented a new method to obtain an exact minimum ESOP for an n variable function by using modified reduced covering function. We also presented various techniques to reduce the computation time and memory requirements. By using this approach, we minimized many ESOPs with $n = 5$ and some ESOPs with up to $n = 9$ variables. Because minimum ESOPs for up to $n = 5$ can be obtained very quickly by a table look-up method, the proposed method is suitable for the functions with $n = 6$ or more. It is possible to extend RCF to treat multiple-output functions. We successfuly minimized adr2 (two bit adder) and mlp2 (two bit multiplier) within 18 minutes. Although we have made a drastic improvement over the previous approach [19], it is still memory and time consuming when the functions require many products.

Table 8.1: ESOP minimization using RCF

n	t	r	# of nodes		CPU
			max	final	(sec)
5	5	2	9069	7	7.0
	6	2	28076	14	12.9
	7	4	84240	861	68.4
	8	4	120032	3732	93.5
	9	4	247440	21887	159.5
6	3	3	8772	11	7.2
	4	3	17656	12	14.1
	5	2	37123	25	23.9
	6	2	114080	40	75.0
	7	2	369273	354	283.5
7	3	2	28062	11	18.1
	4	3	57118	12	68.4
	5	3	218033	21	170.3
	6	2	581322	34	610.2
8	3	2	53832	5	111.6
	4	2	173994	5	705.4
	5	2	904411	7	1483.8
9	3	2	159504	5	796.3
9	4	2	564258	14	3645.4

n : number of input variables
t : number of products in the minimum solution
r : parameter in Theorem 3.1
max : Maximal number of nodes used during computation.
final : Number of nodes in the final BDD for MRCF.
CPU time : HP 9000 Model 720 Workstation 64 MB main memory.

Acknowledgments

This work was supported in part by a Grant in Aid for Scientific Research of the Ministry of Education, Science and Culture of Japan. Mr. M. Matsuura developed the software and did experiments. Prof. N. Koda provided the program for lower bounds on the number of ESOPs. Prof. Jon T. Butler carefully read the manuscript. Mr. S. Minato's comment on the Zero-suppressed BDD is also acknowledged.

References

[1] K. S. Brace, R. L. Rudell and R. E. Bryant," Efficient implementation of a BDD package," *Proc. 27th Design Automation Conference*, June 1990, pp. 40-45.

[2] D. Brand and T. Sasao, "Minimization of AND-EXOR expressions using rewriting rules,"*IEEE Transactions on Computers*, (to be published).

[3] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. Comput.* Vol. C-35, No.8, Aug. 1986, pp.677-691.

[4] C. Damm, "How much EXOR improve on OR," IFIP WG 10.5 Workshop on Applications on the Reed-Muller Expansion in Circuit Design, Sept. 1993.

[5] M. Davio, J-P Deschamps, and A. Thayse, *Discrete and Switching Functions*, McGraw-Hill International, 1978.

[6] S. Even, I. Kohavi and A. Paz, "On minimal modulo-2 sum of products for switching functions," *IEEE Trans. on Electron Computers*, Vol. EC-16, pp.671-674, Oct. 1967.

[7] H. Fleisher, M. Tarvel, and J. Yeager, "A computer algorithm for minimizing Reed-Muller canonical forms," *IEEE Trans. on Computers*, Vol.C-36, No.2, Feb. 1987.

[8] D. H. Green and I. S. Taylor: "Multiple-valued switching circuit design by means of generalized Reed-Muller expansions," *Digital Processes*, vol.2, pp.63-81, 1976.

[9] M. Helliwell and M. Perkowski, "A fast algorithm to minimize multi-output mixed-polarity generalized Reed-Muller forms," *Proc. of the 25th Design Automation Conference*, pp.427-432, June 1988.

[10] N. Koda and T. Sasao, "An upper bound on the number of product terms in AND-EXOR minimum expressions," (in Japanese), *Trans. IEICE*, Vol. J75-D-I, No.3, pp. 135-142, March 1992.

[11] N. Koda and T. Sasao, "A minimization method for AND-EXOR expressions using lower bound theorem," (in Japanese), *Trans. IEICE*, Vol.J76-D-I, No.1, pp. 1-10, Jan. 1993.

[12] N. Koda and T. Sasao, "LP equivalence class of logic functions," IFIP10.5 Workshop on Application of the Reed-Muller expansion in Circuit Design, Sept. 1993.

[13] B. Lin and F. Somenzi, "Minimization of symbolic relations," *Proc. of the IEEE International Conference on Computer Aided Design*, pp.88-91, Nov. 1990.

[14] S. Minato, N. Ishiura, and S. Yajima, "Shared binary decision diagram with attributed edges for efficient Boolean function manipulation," *Proc. 27th ACM/IEEE Design Automation Conf.*, June 1990, pp. 52-57.

[15] S. Minato, "Zero-suppressed BDDs for set manipulation in combinatorial problems," *Proc. 30th Design Automation Conference*, June 1993, pp. 272-277.

[16] A. Mukhopadhyay and G. Schmitz, "Minimization of Exclusive OR and logical Equivalence of switching circuits," *IEEE Trans. Comput.*, C-19, pp. 132-140, 1970.

[17] S. Muroga, *Logic Design and Switching Theory*, John Wiley & Sons, 1979.

[18] G. Papakonstantinou, "Minimization of modulo-2 sum of products," *IEEE Trans. Comput.*, C-28, pp. 163-167, 1979.

[19] M. Perkowski and M. Chrzanowska-Jeske, "An exact algorithm to minimize mixed-radix exclusive sums of products for incompletely specified Boolean functions," *Proc. ISCAS*, pp. 1652-1655, June 1990.

[20] U. Rollwage, "The complexity of mod-2 sum PLA's for symmetric functions," IFIP WG 10.5 Workshop on Applications on the Reed-Muller Expansion in Circuit Design, Sept. 1993.

[21] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, *Field Programmable Gate Arrays*, Kluwer Academic Publishers, Boston 1992.

[22] K. K. Saluja and E. H. Ong, "Minimization of Reed-Muller canonic expansion," *IEEE Trans. Comput.*, C-28, pp. 535-537, 1979.

[23] T. Sasao and P. Besslich, "On the complexity of MOD-2 Sum PLA's," *IEEE Trans. on Comput.* Vol. 39. No. 2, pp. 262-266, Feb. 1990.

[24] T. Sasao, "Logic synthesis with EXOR gates," in (Sasao e.d.) *Logic Synthesis and Optimization*, Kluwer Academic Publishers, 1993, pp.259-285.

[25] T. Sasao, "AND-EXOR expressions and their optimization," in (Sasao e.d.) *Logic Synthesis and Optimization*, Kluwer Academic Publishers, 1993, pp.287-312.

[26] T. Sasao, "EXMIN2:A simplification algorithm for exclusive-OR-Sum-of-products expressions for multiple-valued input two-valued output functions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (to be published).